CheriBSD: a research fork of FreeBSD

Brooks Davis
SRI International

BSDCan, Ottawa, Canada
June 12, 2015
Banks lose over $300m

Office of Personnel Management hacked

80 million customer records
• Compartmentalization decomposes software into isolated components.
• Each sandbox runs with only the rights required to perform its function.
• This model implements the principle of least privilege.
Capsicum

- Hybrid capability model: OS APIs for application compartmentalization
- Out-of-the-box in FreeBSD 10.0
- Growing number of FreeBSD programs are using Capsicum out-of-the-box: tcpdump, auditdistd, hastd, etc.
- Casper framework offers services to sandboxes (e.g., DNS, socket server)
- Google has published a Linux port
From compartments to objects

- Sharing requires pointers with enforced bounds and permissions
- Can we use this mechanism for every pointer?
### CHERI capabilities

- **Unforgeable**
- **Monotonic length and permissions**
- **Tagged memory protects capabilities**
- **Checks apply only on dereference**

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>[64]</td>
</tr>
<tr>
<td>Length</td>
<td>[64]</td>
</tr>
<tr>
<td>Permissions</td>
<td>[32]</td>
</tr>
<tr>
<td>Type</td>
<td>[24]</td>
</tr>
<tr>
<td>Reserved</td>
<td>[8]</td>
</tr>
<tr>
<td>Offset</td>
<td>[64]</td>
</tr>
</tbody>
</table>
C language support

Hybrid:

• \texttt{__capability} annotations on pointers
• Small changes in the C runtime

Pure:

• Compiler compiles code with all pointers are capabilities
• Small application changes to maximize memory safety
Binary compatibility

- **n64**: Pure MIPS
- **Hybrid**: Some pointers are capabilities
- **Pure-capability**: All pointers are capabilities

More compatible → More safe
Hybrid capability/MMU OSes

- **Legacy application + capability libraries**
  - zlib
  - libssl
  - Address-space executive
  - OS kernel
  - CHERI CPU

- **Pure-capability application**
  - zlib
  - libssl
  - Address-space executive
  - Single address space

- **Capability-based OS with legacy libraries**
  - class1
  - class2
  - libssl
  - Address-space executive
  - Single address space
The prototype CPU

- 64-bit MIPS-compatible ISA (≈R4000)
- CHERI ISA extensions
- Runs at 100MHz on FPGA
- Full software stack
CheriBSD supports CHERI

- Platform support (BERI CPU)
- Support for new ISA features
- Infrastructure for compartmentalization
- Custom and adapted applications
- Build system improvements
Lots of deltas!

FreeBSD adapted for the CHERI CPU. WARNING: some programs contain deliberate vulnerabilities
http://www.cl.cam.ac.uk/research/security/ctsr/cheri.html — Edit

This branch is 5988 commits ahead, 620 commits behind freebsd:master

When CPU_CHERI_CSETBOUNDS is used, force use of the CSetBounds even on...

rwatson authored a day ago

latest commit a23a8b6b32
## Kernel changes

<table>
<thead>
<tr>
<th>Component</th>
<th>Files</th>
<th>Lines +</th>
<th>Lines -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headers</td>
<td>19</td>
<td>1424</td>
<td>11</td>
</tr>
<tr>
<td>CHERI initialization</td>
<td>2</td>
<td>49</td>
<td>4</td>
</tr>
<tr>
<td>Context management</td>
<td>2</td>
<td>392</td>
<td>10</td>
</tr>
<tr>
<td>Exception handling</td>
<td>3</td>
<td>574</td>
<td>90</td>
</tr>
<tr>
<td>Memory copying</td>
<td>2</td>
<td>122</td>
<td>0</td>
</tr>
<tr>
<td>Virtual memory</td>
<td>5</td>
<td>398</td>
<td>27</td>
</tr>
<tr>
<td>Object capabilities</td>
<td>2</td>
<td>883</td>
<td>0</td>
</tr>
<tr>
<td>System calls</td>
<td>2</td>
<td>76</td>
<td>0</td>
</tr>
<tr>
<td>Signal delivery</td>
<td>3</td>
<td>327</td>
<td>71</td>
</tr>
<tr>
<td>Process monitoring/debugging</td>
<td>3</td>
<td>298</td>
<td>0</td>
</tr>
<tr>
<td>Kernel debugger</td>
<td>2</td>
<td>264</td>
<td>0</td>
</tr>
</tbody>
</table>
libc changes

• Capability aware memcpy, memmove, etc
• Explicit capability forms of mem* and and str* functions (memcpy_c, memcpy_c_fromcap, memcpy_c_tocap)
• Fixing optimizations based on assumptions about pages
• Split of syscalls and libc (coming soon!)
libcheri

- Compartment object management
- Type allocator
- Loader and runtime linker
- System call implementation for compartments
/usr/lib/libcheri

- Similar to /usr/lib32
- Builds all libraries in pure-capability mode
- Allows for pure-capability programs on a MIPS64 system
Demo Applications

Exploit Mitigated
CHERI prevented an exploit from gaining control!
Tcpdump changes

- Memory safety
- Capability offsets
- Per-protocol
- Pure-capabilities
- Tcpdump 4.6.2
- Linker support

- Added
- Removed
Infrastructure

• Build system improvements
  • Unprivileged builds
  • Per-program (and per-file) compiler replacement
  • Strip during build, not at install
  • …
FreeBSD Journal
http://freebsdjournal.org

Porting FreeBSD to a new CPU, even within a previously supported family, is a significant undertaking.
Early days: Perforce

Pros

• FreeBSD infrastructure
• Good merging
• Easy to maintain stacked branches
• Familiar to team

Cons

• No public access
• Hard to add users
• Not ideal for CI
• Minimal offline support
Perforce ⇒ Github

• Switched October 2013
• Lost some history granularity
• Easy public access
• Trial by fire with git-at-scale
Github model

- Forked freebsd/freebsd repo
- Weird effect: forking CheriBSD seems to fork FreeBSD
- All commits to master branch
- Merge changes from FreeBSD upstream
Merging: first attempt

git fetch upstream

```bash
git merge upstream/master
```

- Merges everything at once!
- Works
- Rebase usually produces insane results
  - Don’t lose the push race!
Oops, we merged a bug!
Bisect is useless
mergify

- Merge one commit at a time
- Mostly true assumption that commits are complete features
- Stream of small changes merging upstream and cheribsd
- Bisect is possible
mergify

• Problem: merging tcpdump went weird
  • Vendor commits have the empty repo as a common parent with master!
  • Solution: merge only direct commits
mergify Demo
Git rebase is broken

• Changes are reapplied in order
• Including merges from vendor branches!
• mergify doesn’t fix this (yet)
• May be an issue with using git wrong or git-svn not handling vendor branches well
mergify TODOs

- rebase mode
- bisect mode
- check that things build/work at key points
Upstreaming

• Reduce merge conflicts

• What to upstream?
  • Drivers for things people can use
  • General infrastructure
  • Infrastructure shared by multiple external consumers
  • Low impact things that are conflict prone
What we’ve upstreamed

- FDT support for MIPS
- Drivers and driver improvements
- Working floating point support for MIPS
- Boot loaders for MIPS
- Unprivileged builds and installs
Related Upstreaming

• Improvements to external projects:
  • QEMU: FreeBSD MIPS64 usermode
  • MIPS64 and ARM packages!
  • Clang/LLVM: MIPS64 fixes
  • LLDB: FreeBSD improvements, MIPS64
  • Tcpdump: better compartmentalization interfaces
Releases

• Internal snapshots
• Restricted releases
• Public releases: http://cheri-cpu.org/
• Shared make-based build infrastructure
Tips for developers

http://xkcd.com/303/ (CC BY-NC 2.5)
Tip 1: Use a big machine

- Enough RAM to hold source and output in cache
  - 128GB is enough for most people
- Fast disk
  - ZFS mirror with large L2ARC and ZIL on flash
- Enough cores
  - 32 on our system
Tip 2: Use a notification service

- I use pushover.net for notifications
- Simple RESTful interface
  - Notifications to iOS and Android devices
  - Also via browser
- Used with a command wrapper script
  
  $ command-notice sleep 60
Tip 3: Build in tmux

• Switch away from running build

• Sending, buffering, and rendering output just to throw it away wasteful

• Even locally, buffering adds delay between end of compilation and control of the terminal
Tip 4: Continuous integration

- Full OS builds after each change or compiler update (out of tree compiler)
- CHERI, MIPS64, and AMD64
- Daily release builds
  - Release kernels booted on hardware and in simulation
- Additional Jenkins jobs build release branches daily
Papers and reports


Future work

- Pure-capability FreeBSD
  - Run legacy MIPS64 code in sandboxes
- CHERI in the kernel
- 128-bit capabilities
- Non-MIPS architectures
Q & A

- October 2011: Deimos capability microkernel runs first sandbox on FPGA prototype
- July 2012: LLVM generates Cheri ISA code
- November 2012: Sandboxed user code runs on CheriBSD; trojan mitigated
- June 2012: CheriBSD capability context for user threads
- December 2013: OS CCAll/CRenumber prototype supports lightweight compartmentalization
- February 2013: SSHed into a Cheri system for the first time
- January 2014: CheriBSD now uses Cheri Clang/LLVM
- November 2014: compartmentalized tcpdump performs multiple domain switches per packet
- October 2010: Project start
- November 2011: tPad FPGA tablet microkernel demo
- May 2012: DE4 FPGA tablet runs FreeBSD
- April 2013: multi-FPGA CheriCloud online for SRI/Cam users
- July 2014: Multicore bringup in progress; capability-offset (‘cursor’) prototype; advanced application prototyping
- May 2015: 128-bit compressed-capability prototype boots